[illegible]

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/899,868	07/05/2001	Jeanne L. Doyle	79-01-003	3339

7590 07/02/2004

David G. Wille, Esq.
Baker Botts L.L.P.
Suite 600
2001 Ross Avenue
Dallas, TX 75201-2980

EXAMINER	
VU, TUAN A	
ART UNIT	**PAPER NUMBER**
2124	

DATE MAILED: 07/02/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

[illegible]

Office Action Summary	Application No.	Applicant(s)
	09/899,868	DOYLE ET AL.
	Examiner	Art Unit
	Tuan A Vu	2124

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE _3_ MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on _05 July 2001_.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) _1-37_ is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) _1-37_ is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on _05 July 2001_ is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _2003/12/31_.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the application filed July 5, 2001.

Claims 1-37 have been submitted for examination.

Claim Rejections - 35 USC § 112

2. The following is a quotation of the second paragraph of 35 U.S.C. 112:

The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

3. Claim 23 is rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Claim 23 recites the element 'the parser' (line 9, 11). There is insufficient antecedent basis for this limitation in the claim. This will be interpreted as 'a parser' for examination of the claim merits.

Claim Rejections - 35 USC § 103

4. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

5. Claims 1-7, 13-18, 23-26, and 37 are rejected under 35 U.S.C. 103(a) as being unpatentable over IBM, "Software Compiler for Analysis and Measuring Programs", 9/1993 (hereinafter Scamp), in view of Leonard, USPN: 5,729,746 (hereinafter Leonard), and further in view of Duan et al., USPN: 6,529,865 (hereinafter Duan).

As per claim 1, Scamp discloses a source code counting system, comprising:

a computer readable storage medium and a software counting tool stored therein and operable to parse a first file containing computer source code to create a token stream in response to one of a plurality of sets of configuration data (e.g. *grammar file* - pg. 123, bottom; *TOKEN COLLECTOR, ...specified in the grammar* - pg. 124-126),

wherein the computer source code file was written in one of a plurality of computer languages that may be processed for the software counting tool (e.g. *independently .. code is written* – pg. 127, 2nd para – Note: a marketable tool to gather metrics necessarily discloses that a variety of programming language can be treated),

create a list of code statements structures in response to the token stream (e.g. *into data structures* – pg. 123 bottom; *token flag* - pg. 124-125), and generate a metric value in response to the list of statements(e.g. *source code measurements, how many times* – pg. 127, 2nd para).

But Scamp does not explicitly disclose that the source code counting system is a source code line counting system, nor does Scamp disclose generating a list of statements in response to the token steam and a count value in response to the list of statements. Analyzing source code to generate a count for a source code, e.g. metrics on size of code generated, being processed was a known concept in the art at the time the invention was made, e.g. LOC from Resource Standard Metrics as submitted in the application IDS. Further, Leonard, in a system using token analyzer analogous to the use of lexer/parser scheme by Scamp, discloses tokens-based measurements and lines of code derived from the token analysis and count for the lines of code derived from such measurement (e.g. col. 4, line 9 to col. 5, line 34). In case Scamp's software measuring system does not already provide a line of code counting tool, it would have been obvious for one of ordinary skill in the art at the time the invention was made to implement such parser-based system by Scamp so that line of code technique as suggested by Leonard can be added because line of codes can be a factor determining how to predict resources, maintain, manage, or improve the state of a software product according to Leonard's approach of having a database for a life-cycle process associated with configuration management activity.

Nor does Scamp explicitly disclose that each set of configuration data to create token stream comprises keywords for one or more of the plurality of computer languages. But in view of the grammar file teachings by Scamp (*TOKEN COLLECTOR, ...specified in the grammar* - pg. 124-126 - Note: a grammar file necessarily entails the understanding of keywords in order to resolve a string of tokens), the grammar file being used per input source code and comprising keywords for the input language to enable token resolution is implicitly disclosed. As for the limitation that one set of configuration data is used among a plurality of configuration data for one or more of a plurality of programming languages, the use of a dictionary for tokenizing a source language of a given programming language was a known concept in the compiler art at the time the invention was made. Duan, in method to provide grammar programming language to any input source code, and using lexical analysis and parser analogous to Scamp and token-based for code generation analogous to Leonard, discloses the use of a dictionary for a given source input file in order to create the lexical baseline and grammar for that particular input source language; hence suggests more than one dictionaries being used for more than one of programming languages (e.g. Fig. 2A and related text; col. 4, lines 11-26). It would have been obvious for one of ordinary skill in the art at the time the invention was made to implement the token collection system by Scamp (combined with Leonard) so that for each input source of a given language, a configuration data set like the dictionary as suggested by Duan can be applied because this way it would alleviate the burden of overloading one single configuration data so to provision for more than one programming language; and would expedite the process of token creation and code lines metrics gathering.

As per claim 2, official notice is taken that at the time the invention was made source code being in programming language being in C++ or Pascal, Fortran, Java was a well-known concept for code analysis and measurement like that disclosed by Scamp. Therefore, it would have been obvious for one of ordinary skill in the art at the time the invention was made to implement to make known programming languages like C++ or Cobol as one of languages being subject for metrics collecting as suggested by Scamp because the purpose to collect metrics to a variety of languages would make the code measuring product as intended by Scamp more attractive.

As per claim 3, Scamp does not explicitly disclose a different set of configuration data from programming languages like C, C++, Java, Cobol, Basic or Fortran; but in view of the teachings by Duan to make the grammar parsing tool a multi-language applicable product and the intention by Scamp to implement the measuring tool onto a variety of languages as in claim 2, the providing of set of configuration data, e.g. dictionary, for more than one type of programming language, e.g. C, C++, Fortran, Java, Cobol or Basic would also have been obvious using the corresponding rationale as set forth both in claim 1 and 2.

As per claim 4, the use of token based on some dictionary or grammar configuration data as disclosed by the combination Scamp/Leonard/Duan from above inherently differentiate tokens in different types represented by a value to the parser in order to enable correct syntax checking (e.g. parse tree) according to the language particular grammar rules; and this concept was a known concept in the art of compiler at the time the invention was made.

As per claim 5, Scamp discloses token type being a comment, an keyword or an identifier, or an operator (e.g. comment, identifier, keyword – from pg.; 124, 6th para to pg. 125, middle; Operator – pg. 127).

As per claim 6, a dictionary or a grammar configuration file assigning a type value inherently teaches maintaining such value consistent with the grammar specific to a particular language; and in view of the teachings by Scamp combined with Duan in claim 1 to impart a specific configuration data file(e.g. one dictionary per language) comprising specification for token type and token type value (re claim 4), the limitation of token type being invariable within one language grammar specification is also disclosed and obvious for the same rationale as to be able to make the measuring tool applicable for more than one language.

As per claim 7, the skipping of token not recognizable by a set of grammar rules during the parsing of token streams as taught by Scamp (e.g. skip over comment token or code structures written by natural language instead of programming language s – re claim 5) discloses portions of the token streams being ignored in generating the statement list because such portions are not in the language among the plurality of computer languages.

As per claim 13, Scamp discloses a method of code counting, comprising:

selecting one set of configuration data (e.g. *grammar file* - pg. 123, bottom – Note: a grammar is specific to one of the languages available, hence submitting one set reads on selecting one among possible other sets for other languages),

wherein collectively such plurality of sets are associated with a plurality of computer languages (e.g. *independently .. code is written* – pg. 127, 2nd para),

wherein such selected set comprises the keyword of a first computer language (e.g. *TOKEN COLLECTOR, ...specified in the grammar* - pg. 124-126 - Note: a grammar file necessarily entails the understanding of keywords in order to resolve a string of tokens),

parsing a first file in the first language into a first token stream in response to the selected set of configuration data (e.g. *TOKEN COLLECTOR, ...specified in the grammar* - pg. 124-126);

generating code constructs and measurement data in response to the token streams analysis (e.g. *into data structures* – pg. 123 bottom; *token flag* - pg. 124-125; *source code measurements, how many times* – pg. 127, 2nd para).

But Scamp does not explicitly disclose lines of code counting, nor does Scamp disclose generating a list of statements in response to the token steam and a count value in response to the list of statements. But these limitations have been addressed in claim 1 from above.

Nor does Scamp explicitly disclose selecting among a plurality of sets of configuration data, each associated with a programming language; but this limitation has been addressed in claim 1 above using Duan's teachings.

As per claims 14-18, these claims correspond to claims 2, 4-6, and 3 respectively, and are rejected using the corresponding rejection as set forth therein.

As per claim 23, Scamp discloses a computer readable medium storing a software counting tool, comprising:

a configuration file for one or more of a plurality of computer languages (e.g. *grammar file* - pg. 123, bottom - Note: a grammar file is specific to possibly one of the languages available, hence submitting one file reads on selecting one among possible other files for other computer languages);

a tokenizer to parse a first language source code into a token stream(*TOKEN COLLECTOR, ...specified in the grammar* - pg. 124-126);

wherein a parser parses source code written in any of the plurality of computer languages (e.g. Figure on pg. 123; *independently .. code is written* – pg. 127, 2nd para);

wherein the parser creates a token stream in response to the configuration file associated with the first computer language(e.g. *TOKEN COLLECTOR, ...specified in the grammar* - pg. 124-126);

generating code constructs and measurement data in response to the token streams analysis (e.g. *into data structures* – pg. 123 bottom; *token flag* - pg. 124-125; *source code measurements, how many times* – pg. 127, 2nd para).

But Scamp does not disclose source code line counting system nor does Scamp disclose a first statement builder to create a list of statements in response to the token steam and a counter to generate a value in response to the list of statements. But these limitations have been addressed in claim 1 from above.

Nor does Scamp explicitly disclose a plurality of configuration files, each associated with one or more of a plurality of computer languages; but this limitation has been addressed in claim 1 above using Duan's teachings.

As per claim 24, Scamp in combination of Leonard and Duan, discloses a plurality of additional statement builders each associated with one or more computer languages and operable to generate a statement list in response to a token stream generated from a source file associated with one or more languages (Note: if Scamp is providing a tool to cooperate with a plurality of languages, the creation of more set of code statements per grammar file (in view of Leonard's teachings) in conjunction with a token stream derived therefrom would be implicitly disclosed as per the rationale of claim 1).

As per claims 25-26, refer to claims 4 and 6, respectively.

As per claim 37, Scamp discloses a code counting system, comprising:

a set of configuration data associated with at least one computer language (e.g. *grammar file* - pg. 123, bottom), a plurality of which set collectively associated with different computer languages (e.g. *independently .. code is written* – pg. 127, 2nd para);

a computer readable medium storing a software counting tool operable to

receive a source code file (Figure pg. 123); and

compute a statistical measure (e.g. *statistics and metrics* - pg. 123) related to the source file using one set of configuration data associated with the computer language that the source code was written in (e.g. *syntax for a particular programming language* - pg 125, bottom- pg. 126).

But Scamp does not disclose source code line counting system nor does Scamp disclose a statistical measure related to the number of source code lines in response using the one of the sets configuration data. But these limitations have been addressed in claim 1 from above using Leonard.

But Scamp does not explicitly disclose a plurality of set of configuration data. But this limitation has been addressed in claim 1 using Duan.

6. Claims 8-11, 19-21, and 27-28 are rejected under 35 U.S.C. 103(a) as being unpatentable over IBM, “Software Compiler for Analysis and Measuring Programs”, 9/1993 (i.e. Scamp), in view of Leonard, USPN: 5,729,746 and Duan et al., USPN: 6,529,865, as applied to claims 1, 13, 17, 25, and further in view of Greenfeld, USPN: 4,931,928 (hereinafter Greenfeld).

As per claims 8-11, validating code constructs by identifying them by correct sequences being identifiable via a parsing process was a well-known concept at the time of the invention, i.e. the use of a parser to segregate tokens in valid and identifiable sequences for a given grammar was an inherent process in compilers when the token streams is analyzed using a parse tree so as to validate the constructs of the source code with such constructs being assigned with specific identification. Thus, the teachings of a parser by Scamp/Leonard or Duan implicitly discloses (re claim 8) statement being categorized in different types and that (re claim 11) a relationship between tokens in a parse tree is examined with respect to other tokens in the stream when the code statement is created and verified for correctness. The assigning of a type value for each code statement type would also have been obvious in light of the well-known concepts and the rationale as set forth in claim 5 from above.

But Scamp does not explicitly teach that the statement type value does not vary based on the computer language of the source code of the first file (re claim 9), that the type is either data, executable or compiler (re claim 10). The rationale as to maintain a statement type to be invariable within a grammar rule sets of a specific programming language would have been obvious herein using the same rationale as set forth in claim 6 above. Further, Greenfeld, in a method to provide reuseable set of language configuration data for program code analysis analogous to the grammar file by Scamp or to the dictionary by Duan, discloses Semantics information enabling extraction of code constructs and differentiation the type being extracted as code symbols, control flow information or data and relationships information between type (e.g. col. 5, line 30-44). It would have been obvious for one of ordinary skill in the art at the time the invention was made to implement the differentiation of code statement in type as suggested by Scamp/Leonard/Duan's parsing process such that the type can be code, data, and compiler control directives as suggested by Greenfeld because this way code statements can be distinguished easily so to make efficient runtime resources allotment and also can be persisted based on its category or type enabling a database of information to be reused for further analysis as mentioned by Greenfeld.

As per claims 19-20, these claims correspond to claims 8 and 9, respectively and are rejected using the corresponding rejection as set forth therein.

As per claim 21, this claim includes the limitations of claim 8 and 9, from above, and is rejected using the corresponding rejections as set forth therein.

As per claims 27-28, these claims correspond to claims 8 and 9, respectively and are rejected using the corresponding rejection as set forth therein.

7. Claims 12 and 22 are rejected under 35 U.S.C. 103(a) as being unpatentable over IBM, “Software Compiler for Analysis and Measuring Programs”, 9/1993 (i.e. Scamp), in view of Leonard, USPN: 5,729,746, Duan et al., USPN: 6,529,865, and Greenfeld, USPN: 4,931,928, as applied to claims 1, 13, and further in view of Bloom, USPN: 3,711,863 (hereinafter Bloom).

As per claim 12, Scamp does not disclose creating a second list of statements from the token stream generated from parsing a second source file of a different version than the first source file; and comparing the first list of statements to the second list of statements. However, code being developed within a life-cycle encompassing code change management is suggested by Leonard (Fig. 1-3); and the merging of code based on a baseline code to extract differences among other developer’s code instances is suggested by Greenfeld (e.g. col. 7, lines 35-53). Further, Bloom, in a method teaching code comparator technique such as that suggested by Greenfeld or any code merging in software configuration management, discloses comparing two files to extract the difference and save the number of delta lines (e.g. step 54 - Fig. 2C). It would have been obvious for one of ordinary skill in the art at the time the invention was made to manage the code analysis and software development by Scamp/Leonard in view of Greenfeld’s code difference extraction, so as to implement a second list of statements as from a token stream derived from a second file, a delta extraction of code differences as taught by Bloom, and record a count responsive to the differences from comparing the first list of statements and the second list of statements according to the scheme of token generation as addressed in claim 1. The motivation is that this would enable to store a difference between code versions in terms of a value which can be persisted according to

Greenfeld's baseline persisting method and life-cycle code analysis by Scamp/Leonard, thus providing numerical representation of the difference as taught by Bloom between two versions of code being managed under the management tool by Scamp/Leonard (enhanced by the baseline code comparing by Greenfeld).

As per claim 22, refer to rejection of claim 12.

8. Claims 29-36 are rejected under 35 U.S.C. 103(a) as being unpatentable over IBM, "Software Compiler for Analysis and Measuring Programs", 9/1993 (i.e. Scamp), in view of Leonard, USPN: 5,729,746, and Greenfeld, USPN: 4,931,928; and further in view of Bloom, USPN: 3,711,863 (hereinafter Bloom).

As per claim 29, Scamp discloses a measuring method comprising parsing a first file to create a token stream, creating code structures in response to the stream (refer to rejection of claim 1); but does not teach creating a first list of statements in response to the token stream using Leonard. But such limitation has been addressed in claim 1 above.

Nor does Scamp disclose creating a second list in response to a second token stream from parsing a second file; and comparing the first list to the second list of statements to generate one count responsive to such comparison. The teachings to provide software analysis with LOC count generating in a life-cycle code management is taught by Leonard and the providing of code baseline for merging or identifying code version differences has been taught by Greenfeld (re claim 12). Further, the limitation as to compare files to generate a count based on the differences between the files would have been obvious in view of Bloom's counting of delta lines as set forth in the rationale in claim 12 above combined with the teachings of the combined Scamp/Leonard/Greenfeld.

As per claims 30-33, merging tools used to exhibit the differences between 2 files set side-by-side implicitly disclose accounting for the lines of source code that are unchanged or changed for each file being submitted in the merging tool, and this concept was a well-known concept at the time the invention was made. Using the teachings by Bloom to count the differences, the limitations as to account for the number of unchanged/changed lines in one file with respect to the other file among the files being compared would have been but one of the variations from making use of the results out of the delta extraction as suggested in the counting by Bloom or many file merging tools. Hence, it would have been obvious for one of ordinary skill in the art at the time the invention was made to implement the merging tool as suggested by Greenfeld or the comparator by Bloom so that (re claim 30)the number of statements modified in the 2nd file with respect to the first file, (re claims 31-32) the number of statements in one file but not in the other, and (re claim 33) the number of statements commonly present in both files would also be generated or computed depending on the needs for gathering information representing how the changes to the files are to be persisted and recorded for future analysis, according to the software configuration management or versioning so well-known as suggested by Leonard or Greenfeld.

As per claim 34, only Greenfeld suggests comparing against a modified version of the first file (e.g. col. 7, lines 35-53); and the limitations in this claim are the same as recited in the claims 30-33, respectively. In light of the rationale used in the rejection from claims 30-33, this claim is also rejected based on the corresponding rejection as set forth therein.

As per claim 35, this claim is a computer-readable medium claim of claim 29 above, hence is rejected using the rejection as set forth therein.

As per claim 36, this claim corresponds to claim 34, hence is rejected using the same rationale as set forth therein.

Conclusion

9. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (703)305-7207. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (703)305-9662.

Any response to this action should be mailed to:

Commissioner of Patents and Trademarks

Washington, D.C. 20231

or faxed to:

(703) 872-9306 (for formal communications intended for entry)

or: (703) 746-8734 (for informal or draft communications, please label "PROPOSED" or "DRAFT" – please consult Examiner before use)

Hand-delivered responses should be brought to Crystal Park II, 2121 Crystal Drive, Arlington. VA. , 22202. 4th Floor(Receptionist).

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703) 305-3900.

VAT
June 27, 2004

[illegible]

KAKALI CHAKI
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100